IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Examiner: Kang, Insun
Xinliang David Li, et al.	
Application No.: 10/758,376	Art Unit: 2193
Filing Date: January 15, 2004	Attorney Docket No.: 200313024-1
Title: Program Optimization	

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF FILED UNDER 37 C.F.R. § 41.37

Sir:

This appeal brief follows the Notice of Appeal mailed by Applicants on March 26, 2008. Per the return postcard, the Notice of Appeal was received by the USPTO on March 31, 2008.

The Commissioner is hereby authorized to charge requisite fees due for this submission to Deposit Account No. 08-2025 (Hewlett Packard).

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas. The Hewlett-Packard Development Company, L.P., is the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

On information and belief, there are no appeals, interferences, or judicial proceedings known to the appellant, the appellant's legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board of Patent Appeals and Interferences (the "Board") decision in the pending appeal.

III. STATUS OF CLAIMS

A. Total Claims: 1-35

B. Current Status of Claims:

1. Claims canceled: 3 and 19
2. Claims withdrawn: none
3. Claims pending: 1, 2, 4-18 and 20-35
4. Claims allowed: none
5. Claims rejected: 1, 2, 4-18 and 20-35
6. Claims objected to: none

C. Claims on Appeal: 1, 2, 4-18 and 20-35

As indicated above, claims 1, 2, 4-18 and 20-35 are pending in this application, stand finally rejected, and are being appealed. These claims are rejected in the final office action mailed January 3, 2008 ("the last office action").

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter relates to program compilation. As described in the application, there are difficulties in using whole program analysis to effectively optimize a program being compiled when the program utilizes object files, archive libraries, and/or shared libraries. The claimed subject matter relates to a method and system to improve optimization achievable by whole program analysis.

Independent claim 1 relates to a method of generating a software program executable binary file. A first file for a first module (specification, page 6, lines 11-24, see files "1.c", "2.c" or "3.c", for example) and a second file for a second module (specification, page 6, lines 11-24, see files "4.o" or "5.o", for example) are accessed. The first file includes source code (specification, page 6, lines 11-24, see files "1.c", "2.c" or "3.c", for example), and the second file includes object code (specification, page 6, lines 11-24, see files "4.o" or "5.o", for example) and object file summary information (specification, page 6, line 25 through page 7, line 6). An executable binary file is generated from at least the first and second files. (Specification, page 6, lines 17-24.) The object file summary information is used in optimizing the executable binary file generated. (Specification, page 2, lines 7-8, and page 5. line 31 through page 6, line 2.) The object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table. (Specification, page 8, lines 7-29.)

Independent claim 26 relates to a system for generating a software program executable file. The system includes a processing device (202 in FIG. 2) configured to execute computer-readable program code, and a memory system (204 in FIG. 2) configured to store the computer-readable program code and data. The system also includes a source file (112 in FIG. 1, and see 104 in FIGS. 1 and 2) for a first module comprising source code stored by the memory system (204 in FIG. 2), an object file (see 120 and 122 in FIG. 1) for a second module including computer-readable program code and object file summary information (specification, page 7, lines 25 through page 8, line 3, for example), and a translator (102 in FIGS. 1 and 2) comprising computer-readable program code stored by the memory system (204 in FIG. 2). The computer-readable program code of the translator (102 in FIGS. 1 and 2) is configured to access at least the source and object files and to generate the executable file of the program therefrom

(specification, page 3, lines 1-4, for example). The object file summary information is used in optimizing the executable file generated. (Specification, page 2, lines 7-8, and page 5. line 31 through page 6, line 2.) The object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table. (Specification, page 8, lines 7-29.)

Independent claim 31 relates to a computer-readable storage medium storing an object file of a computer programming module. The computer-readable storage medium includes computer-readable object code (specification, page 6, lines 11-24, see files "4.o" or "5.o", for example) for the module and computer-readable object file summary information. . (Specification, page 2, lines 7-8, and page 5. line 31 through page 6, line 2.) The computer-readable object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module. (Specification, page 8, lines 7-29.)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are to be reviewed on appeal:

1. The rejection of claims 1, 2, 4-14, 16-18, 20-23, 25-29, 31-33 and 35 under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940).

2. The rejection of claims 15 and 34 under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Haber et al (US Patent 6,966,055).

3. The rejection of claim 24 under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Hiranandani et al (US Patent 5,812,855).

4. The rejection of claim 30 under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Soroker et al (US Patent 6,219,834).

VII. ARGUMENT

Applicants respectfully traverse the aforementioned rejection of claims 1, 2, 4-18 and 20-35 in the last office action for the following reasons.

A. Claims 1, 2, 4-14, 16-18, 20-23, and 25-29

Claims 1, 2, 4-14, 16-18, 20-23, 25-29, 31-33 and 35 stand rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 1 recites as follows.

> 1. A method of generating a software program executable binary file, the method comprising:
> accessing a first file for a first module including source code therein;
> accessing a second file for a second module including object code therein and further including object file summary information; and
> generating the executable binary file from at least the first and second files,
> wherein the object file summary information includes a **summary intermediate representation (SIR)** and an **extension to a linker symbol table**, and
> wherein the **object file summary information** is used in **optimizing the executable binary file** generated.

(Emphasis added.)

As shown above, claim 1 recites that "the **object file summary information** is used in **optimizing the executable binary file** generated." Claim 1 further recites that "the

object file summary information includes a **summary intermediate representation (SIR)** and an **extension to a linker symbol table**". (Emphasis added.)

First, consider the limitation that "the **object file summary information** is used in **optimizing the executable binary file** generated." The use of the object file summary information in optimizing the executable binary file is described in the specification, for example, as follows. "In accordance with an embodiment of the invention, information needed to perform certain types of whole program analysis are determined and saved into the object files. This information may be thought of as being equivalent to having the whole source for those modules available with respect to those types of analysis. We call this information 'object file summary information' (OFSI) to distinguish this type of summary data from the summary data stored in intermediate representation (IR) files which are used for the purpose of speeding up the compile time." (Specification, page 6, lines 26-33.)

In contrast, the latest office action cites to column 12, lines 40-47 in relation to the claim limitation that "the object file summary information is used in optimizing the executable binary file generated." (See the bottom of page 2 of the latest office action.) For convenience of reference, column 12, lines 40-47 of Sato is reproduced below

> The program executing unit 80, after completion of running the temporary object program, **adds up each number of the procedure calling times on each procedure actually called, at every position of indirect call codes and stores the resultant summation** together with the address of each procedure into the second information storing unit 100, as the dynamic information (dynamic information collecting processing 421 in FIG. 4).
>
> (Emphasis added.)

However, the "summation" referred to in the above citation is <u>not</u> object file summary information used in optimizing the executable binary file. Rather, the "summation" is literally a numerical **total** from adding up "each number of the procedure calling times on each procedure actually called, at every position of indirect call codes".

Second, consider the limitation that "the object file summary information includes a **summary intermediate representation (SIR)** ...." As described in the specification,

"The SIRs may also be referred to as per-procedure summary data." (Page 8, lines 11-12.) "The SIR includes a summary symbol table per procedure and a list including exposed pointer assignments (SIR assignments)." (Page 8, lines 28-29.)

In contrast, the latest office action does not appear to cite to any portion of Sato in relation to the claim limitation that "the object file summary information includes a summary intermediate representation (SIR)". There is no mention of which portion of Sato that is being read onto the claimed "summary intermediate representation." See detailed rejection of claim 1 on page 2 and the Response to Arguments on page 10, neither of which appear to recognize this claim limitation.

Third, consider the limitation that "the object file summary information includes ... **an extension to a linker symbol table** ...." As is directly apparent from its name, a linker symbol table is a table of symbols which is used by the linker. More particularly, the linker symbol table is described in the specification on page 8, lines 13-17, as follows. "The linker symbol table exists for relocatable objects and shared libraries (if not stripped). There is one linker symbol table per module or per shared library. The table defines the global context for the summary data. Each linker symbol table entry has a unique index, here referred to as a linker identifier (LI_ID)." An extension to the linker symbol table is described in the specification on page 8, lines 18-25, as follows. "In accordance with this embodiment, the extension to the linker symbol table comprises a boolean flag "doesNotExposeAddress" added for each procedure identified therein. If the procedure does not expose addresses of any memory location, this flag is set to true for that procedure. A procedure does not expose addresses of any memory location when it does not save the address into any memory location accessible outside of the procedure. For all such procedures with the flag set to true, no additional OFSI is needed for the purpose of points-to analysis."

In contrast, the latest office action cites to the "temporary object program" (i.e. col. 12, lines 20-39) of Sato as disclosing the claimed "extension to the linker symbol table". (See latest office action, page 10, under Response to Arguments.) Applicants respectfully point out that a "temporary object program" is not the same as or even similar to an "extension to the linker symbol table." As discussed in Sato, "The code inserting unit 41 compiles the source program (refer to FIG. 2) read from the first

program storing unit 10 to a temporary object program executable by a processor ...." (Col. 12, lines 1-3, emphasis added.) In other words, **the temporary object program is an executable** generated by the compiler. Applicants respectfully submit that there is no disclosure in Sato that the "temporary object program" incorporates a **table of symbols used by the linker**, much less an **extension** to such a table of symbols.

Thus, for at least the above-discussed reasons, applicants respectfully submit that claim 1 overcomes this rejection.

Claims 2. 4-14, 16-18, 20-23, and 25 depend from claim 1. Thus, claims 2. 4-14, 16-18, 20-23, and 25 overcome this rejection for at least the reasons discussed above in relation to claim 1.

Claim 26 is a system claim which recites both that "the **object file summary information** is used in **optimizing the executable file** generated" and that "the object file summary information includes a **summary intermediate representation (SIR)** and an **extension to a linker symbol table**". (Emphasis added.) Thus, claim 26 overcomes this rejection for at least the reasons discussed above in relation to claim 1.

Claims 27-29 depend from claim 26. Thus, claims 27-29 overcome this rejection for at least the reasons discussed above in relation to claim 26.

B. Claims, 31-33 and 35

Claims 31-33 and 35 stand rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 31 is a computer-readable storage medium claim which recites "computer-readable object file summary information including a **summary intermediate representation (SIR)** and an **extension to a linker symbol table** ...." (Emphasis added.) Thus, claim 31 overcomes this rejection for at least the second and third reasons discussed above in relation to claim 1.

Claims 32-33 and 35 depend from claim 31. Thus, claims 32-33 and 35 overcome this rejection for at least the reasons discussed above in relation to claim 31.

C. Claims 2 and 27

Claims 2 and 27 stand rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 2 recites "**disambiguating memory accesses otherwise considered aliased** using the object file summary information." (Emphasis added.) This aspect is discussed in the specification. For example, page 7, lines 8-16, states as follows. "In accordance with an embodiment of the invention, the object file summary information (OFSI) is used in performing a "points-to" analysis so as to disambiguate memory accesses otherwise considered aliased. Points-to analysis determines the points-to relations of memory locations or memory alias information. The results of the points-to analysis may be used by a compiler to disambiguate memory accesses that are otherwise considered aliased. The improved alias information provided by points-to analysis advantageously benefits optimization of the compiled code, including scheduling performed by the compiler backend."

The first office action cited to col. 10, lines 15-24 against claim 2. This citation is reproduced below for convenience of reference.

> The code inserting unit 41 compiles a source program to a temporary object program executable by a processor and stores the object program in the third program storing unit 50. At this time, the code inserting unit 41 inserts a code for supplying to an outside file the identification information of a procedure actually called at a position (for example, the address of the procedure, or the like) when the program executing unit 80 runs the temporary object program, into the corresponding position of an indirect procedure call code.
>
> (Emphasis added.)

As shown above, this citation to Sato relates to "identification information of a procedure actually called" and does not disclose or teach the claimed "**disambiguating memory accesses otherwise considered aliased....**"

The latest office action states in a cryptic manner that "Sato resolves the unresolved references from indirect calls and dynamic information (i.e. col. 12, lines 40-53)." (Page 11, lines 1-2.) However, there is no explanation as to how resolving "unresolved references from indirect calls and dynamic information" pertains at all to the claimed feature of "**disambiguating memory accesses otherwise considered aliased** using the object file summary information."

For convenience of reference, col. 12, lines 40-53 are reproduced below.

> The program executing unit 80, after completion of running the temporary object program, adds up each number of the procedure calling times on each procedure actually called, at every position of indirect call codes and stores the resultant summation together with the address of each procedure into the second information storing unit 100, as the dynamic information (dynamic information collecting processing 421 in FIG. 4). In the example of FIG. 6, the number of the procedure calling times on the procedure of the address "1048784" is the greatest. Therefore, as a result of the dynamic information collecting processing, the number of the procedure calling times on the procedure "g" of the address "1048784" is supposed to be the greatest, at the sixth line of the temporary object program shown in FIG. 5.

As seen above, the citation to col. 12, lines 40-53 of Sato discloses storing a **summation (here, merely meaning a total value)** relating to a number of procedure calls and the address of each procedure. Applicants respectfully submit that this citation does not disclose the claimed feature of "**disambiguating memory accesses otherwise considered aliased** using the object file summary information."

Therefore, for at least the above-discussed reasons, applicant respectfully submits that claim 2 overcomes this rejection.

Claim 27 is a system claim which recites that "a points-to analyzer that uses the object file summary information to **disambiguate memory accesses otherwise considered aliased**." (Emphasis added.) Thus, claim 27 overcomes this rejection for at least the reasons discussed above in relation to claim 2.

D. Claim 4

Claim 4 stands rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 4 recites that "the extension to the linker symbol table includes a **flag indicating whether a procedure exposes a memory address** by storing the address in a location accessible outside the procedure." (Emphasis added.)

Against claim 4, the latest office action states that "Sato discloses that the operator can specify the address of a position of an indirect call procedure code in the object program (i.e. col. 16, lines 1-30)." (Page 11, lines 7-11.) This citation is reproduced below for convenience of reference.

> For example, although the respective program storing units 10, 30, 50, 60, 70, the respective information storing units 20, 100, and the input data storing unit 90 are separately constituted as individual storing medium, in the above-mentioned embodiment, as illustrated in FIG. 1, they are not restricted to this way, but they may be constituted physically in a single storing medium having various storing areas. In this case, since the respective program storing units 10, 30, 50, 60, 70 are to store a program requiring large storing capacity, they may be constituted by an external storing device such as a floppy disk, hard disk, or CD-ROM; since the respective information storing units 20, 100 and the input data storing unit 90 are to store data requiring comparatively smaller storing capacity, they may be constituted by semiconductor memory such as ROM or RAM.
>
> In the above-mentioned embodiment, the code inserting unit 41 is provided in the compiler 40, so to insert a code for supplying the identification

> information of a procedure actually called at a position to an outside file when the program executing unit 80 runs the temporary object program, into the position of indirect call procedure code. Alternatively, for example, a means for collecting information of running a program may be provided in the program executing unit 80, so to collect the procedure identification information in running the temporary object program, instead of providing the compiler 40 with the code inserting unit 41. By way of example, in the case of a software simulator which simulates the operation of a processor by software, the software simulator may be provided with a function such that, at execution time of an object program, an operator can **specify the address of a position of an indirect call procedure code** in the object program, so to supply the value of a variable for indirectly calling a procedure at a position (in the object program used in the above-mentioned embodiment, the variable "fp") to the outside**; thereby, the function makes it possible to supply the address of a procedure to an outside file**.
>
> (Emphasis added.)

As shown above, this citation to Sato discloses that "the software simulator may be provided with a function such that, at execution time of an object program, an operator can **specify the address of a position of an indirect call procedure code** in the object program ...**thereby, the function makes possible to supply the address of a procedure to an outside file**." (Emphasis added.)

Applicants respectfully submit that **specifying an address of a position of an indirect call procedure code so as to be able to supply the address to an outside file** is not the same as, nor is it equivalent to, the claimed "**flag indicating whether a procedure exposes a memory address** ...." Specifying an address for a procedure is *entirely different* from a flag indicating whether a procedure exposes a memory address. The address for a procedure does not indicate whether or not a procedure exposes a memory address.

Therefore, for at least the above-discussed reasons, applicant respectfully submits that claim 4 overcomes this rejection.

E. Claim 9

Claim 9 stands rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Regarding the summary information representation (SIR), claim 9 recites that "the SIR uses an **operator for memory referencing**." (Emphasis added.) As described in detail in the specification, "The deref operator comprises an operator for memory referencing.... The deref operator may take the form of deref (SYMID | address_expression, <offset>). In this form, the deref operator takes an expression or SYMID as the first operand and an optional offset field." (Specification, page 9, lines 28-33.)

In contrast, the citation in the latest office action to col. 12, lines 62-67 of Sato discloses a branch condition code and indirect and direct call codes. No mention of any "operator for memory referencing" is found in that citation.

Therefore, for at least the above-discussed reasons, applicant respectfully submits that claim 9 overcomes this rejection.

F. Claim 10

Claim 10 stands rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 10 recites that "the SIR uses an **operator to adjust an address expression by an offset**." (Emphasis added.) As described in detail in the specification, "The off_adjust operator may be used to adjust the address expression by a certain (non-zero) amount of offset in bytes. The off_adjust operator may take a form that uses two operands, wherein a first operand comprises the expression, and a second operand comprises the offset value." (Specification, page 10, lines 10-14.)

The citation in the latest office action to col. 12, lines 54-62 of Sato discloses reading the number of the procedure calling times on each procedure and judging whether a given procedure call satisfies a predetermined condition. No mention of any "operator to adjust an address expression by an offset," or anything similar thereto, is found in that citation.

Therefore, for at least the above-discussed reasons, applicant respectfully submits that claim 10 overcomes this rejection.

G. Claim 12

Claim 12 stands rejected under 35 U.S.C. §102(b) as being anticipated by Sato (US Patent 6,292,940). Applicants respectfully traverse this rejection.

In order to establish a case of anticipation, the prior art reference must disclose all the claim limitations.

Claim 12 recites that "the SIR uses an **operator to merge pointer values from different control flow paths.**" (Emphasis added.) As described in detail in the specification, "The merge operator comprises the operator used to merge pointer values from different control flow paths. The merge operator is used for copy propagating local variable values to global variables." (Specification, page 11, lines 1-3.)

The citation in the latest office action to col. 12, lines 62-67 of Sato discloses a branch condition code and indirect and direct call codes. No mention of any "operator to merge pointer values", nor anything equivalent thereto, is found in that citation.

Therefore, for at least the above-discussed reasons, applicant respectfully submits that claim 12 overcomes this rejection.

H. Claims 15 and 34

Claims 15 and 34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Haber et al (US Patent 6,966,055). Applicants respectfully traverse this rejection.

Claim 15 depends from claim 1, and claim 34 depends from claim 31. For reasons discussed above in section A, claims 1 and 31 are patentably distinguished over Sato. Haber et al. is cited in relation to a nop instruction and does not cure the deficiencies of Sato in relation to claims 1 and 31.

Since claims 1 and 31 are patentably distinguished over Sato in view of Haber et al, dependent claims 15 and 34 are also patentably distinguished over Sato in view of Haber et al. Thus, claims 15 and 34 overcome this rejection.

I. Claim 24

Claim 24 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Hiranandani et al (US Patent 5,812,855). Applicants respectfully traverse this rejection.

Claim 24 depends from claim 1. For reasons discussed above in section A, claim 1 is patentably distinguished over Sato. Hiranandani et al. is cited in relation to reusing existing code and does not cure the deficiencies of Sato in relation to claim 1.

Since claim 1 is patentably distinguished over Sato in view of Hiranandani et al, dependent claim 24 is also patentably distinguished over Sato in view of Hiranandani et al. Thus, claim 24 overcomes this rejection.

J. Claim 30

Claim 30 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Sato (US Patent 6,292,940) in view of Soroker et al (US Patent 6,219,834). Applicants respectfully traverse this rejection.

Claim 30 depends from claim 26. For reasons discussed above in section A, claim 26 is patentably distinguished over Sato. Soroker et al. is cited in relation to compiler extensions comprising APIs for communication between a compiler and a linker and does not cure the deficiencies of Sato in relation to claim 26.

Since claim 26 is patentably distinguished over Sato in view of Soroker et al, dependent claim 30 is also patentably distinguished over Sato in view of Soroker et al. Thus, claim 30 overcomes this rejection.

VIII. CONCLUSION

For at least the above reasons, applicants respectfully request that the rejections of claims 1, 2, 4-18 and 20-35 be overturned.

Respectfully submitted,
Xinliang David Li, et al.

Dated: May 19, 2008

James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O. Box 641330
San Jose, CA 95164
Tel.: (408)436-2110
Fax.: (408)436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	~~May 14, 2008~~
Express Mail Mailing Number (*optional*):			

CLAIMS APPENDIX

CLAIMS INVOLVED IN THE APPEAL

1. A method of generating a software program executable binary file, the method comprising:
accessing a first file for a first module including source code therein;
accessing a second file for a second module including object code therein and further including object file summary information; and
generating the executable binary file from at least the first and second files,
wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table, and
wherein the object file summary information is used in optimizing the executable binary file generated.

2. The method of claim 1, further comprising disambiguating memory accesses otherwise considered aliased using the object file summary information.

4. The method of claim 1, wherein the extension to the linker symbol table includes a flag indicating whether a procedure exposes a memory address by storing the address in a location accessible outside the procedure.

5. The method of claim 1, wherein the SIR includes a summary symbol table.

6. The method of claim 5, wherein the summary symbol table includes global and static symbols accessed in a procedure, formal parameters of the

procedure, return location for the procedure, and other procedures called by the procedure.

7. The method of claim 6, wherein a symbol is referenced in the summary symbol table by using an associated summary symbol identifier (SYMID).

8. The method of claim 7, wherein a symbol entry includes a linker identifier (LI_ID) of the entry from a linker symbol table.

9. The method of claim 5, wherein the SIR uses an operator for memory referencing.

10. The method of claim 5, wherein the SIR uses an operator to adjust an address expression by an offset.

11. The method of claim 5, wherein the SIR uses an operator to take an address of a function or variable.

12. The method of claim 5, wherein the SIR uses an operator to merge pointer values from different control flow paths.

13. The method of claim 5, wherein the SIR uses an operator to represent direct procedure calls.

14. The method of claim 5, wherein the SIR uses an operator to represent indirect procedure calls.

15. The method of claim 5, wherein the SIR uses a no-operation type operator to discard values.

16. The method of claim 5, wherein the SIR includes a control data structure comprising a link field for each procedure that points to an SIR block of a next procedure.

17. The method of claim 5, wherein the SIR includes a control data structure comprising a table having links to an SIR block for each procedure.

18. The method of claim 1, further comprising determining variables modified by and referenced by function calls in the object code using the object file summary information.

20. The method of claim 18, wherein the extension to the linker symbol table includes a first flag indicative of whether a procedure modifies non-local variables and a second flag indicative of whether the procedure references non-local variables.

21. The method of claim 20, wherein the extension to the linker symbol table includes a second flag indicative of whether the procedure modifies global/static variables excluding callees and a third flag indicative of whether the procedure references non-local variables excluding callees.

22. The method of claim 18, wherein the per-procedure summary data comprises a linked list of entries corresponding to symbols directly modified or referenced in a procedure.

23. (The method of claim 22, wherein each entry comprises a linker identifier of a corresponding symbol and flags indicative of whether that symbol is modified or referenced.

24. The method of claim 1, wherein the second file comprises a load module that is a shared library of procedures.

25. The method of claim 1, wherein multiple files including object code are accessed and used in compiling the program.

26. A system for generating a software program executable file, the system comprising:

a processing device configured to execute computer-readable program code;

a memory system configured to store the computer-readable program code and data;

a source file for a first module comprising source code stored by the memory system;

an object file for a second module including computer-readable program code and object file summary information; and

a translator comprising computer-readable program code stored by the memory system, wherein the computer-readable program code of the translator is configured to access at least the source and object files and to generate the executable file of the program therefrom,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table, and

wherein the object file summary information is used in optimizing the executable file generated.

27. The system of claim 26, further comprising a points-to analyzer that uses the object file summary information to disambiguate memory accesses otherwise considered aliased.

28. The system of claim 26, further comprising a module that uses the object file summary information to determine variables modified by and referenced by function calls in the object file.

29. The system of claim 26, wherein the translator comprises:
a compiler configured to translate source files into intermediate files; and
a linker configured to access the object file summary information and communicate information to the compiler relevant to optimizing compilation of the program.

30. The system of claim 29, wherein the translator further comprises a feedback provider that provides a communications interface between the compiler and the linker.

31. A computer-readable storage medium storing an object file of a computer programming module, the computer-readable storage medium comprising:
computer-readable object code for the module; and
computer-readable object file summary information including a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module.

32. The computer-readable storage medium of claim 31, wherein the SIR includes a summary symbol table.

33. The computer-readable storage medium of claim 32, wherein the summary symbol table includes global and static symbols accessed in the module, formal parameters of the module, return location for the module, and other procedures called by the module.

34. The computer-readable storage medium of claim 31, wherein the SIR uses a plurality of operators from a group of operators including an operator for memory referencing, an operator to adjust the address expression by an offset, an operator to take an address of a function or variable, an operator to merge pointer values from different control flow

paths, an operator to represent direct procedure calls, an operator to represent indirect procedure calls, and a no-operation type operator to discard values.

35. The computer-readable storage medium of claim 31, wherein the SIR includes a linked list of entries corresponding to symbols directly modified or referenced in a procedure.

EVIDENCE APPENDIX

There are no documents or items submitted under this section.

RELATED PROCEEDINGS APPENDIX

There are no documents or items submitted under this section.